The WindBlown Adventure

Design Document

DADIU08 - team 4

Table of Contents

Vision Statement 4
Game Outline 4
Game Synopsis 4
Setting 4
Audio-visual Style 5
Key Features 5
Target Audience 5
Characters 6
Ragga 6
The Inventor 6
Story 7
Synopsis 7
Intro Cut-scene 7
In-game Synopsis 7
Gameplay 8
Overview 8
Level structure 8
Gradually Increasing Difficulty 8
Time Limit 8
Checkpoint System 8
Obstacles 9
Hi-score 10
Avatar Actions and Movement 10
Acquiring Abilities 10
Controls and Control Modes 10
Scissor-legs 10
Health 11
Interface 11
Menu Screen 11
Playscreen Wireframe 11
Ability Select Wireframe 12
Level Design 13

Level 1 – The Dangerous Kitchen 13
Sound Design 15
Kitchen Universe 15
Gameplay 15
The Music 15
Technical Implementation 16
Technical Specifications 17
Platform and OS 17
External Code 17
Modifications 17
Milestone 1: Playable prototype 17
Milestone 2: Beta version 18
Milestone 3: Play-test version 19
Milestone 4: Final Release 19
Other Elements 20
Artificial Intelligence 20
Additional Features 21
Audio-visual Features 21
Non-interactive sequences 21
Character models 21
Cut-scenes 21
Project Estimate (first draft) 22
The Brushy WindBlown Adventure 22

Vision Statement

Game Outline

The game is a side-scrolling 2½D platformer, where the main character can switch between ability sets, in order to navigate the environment.

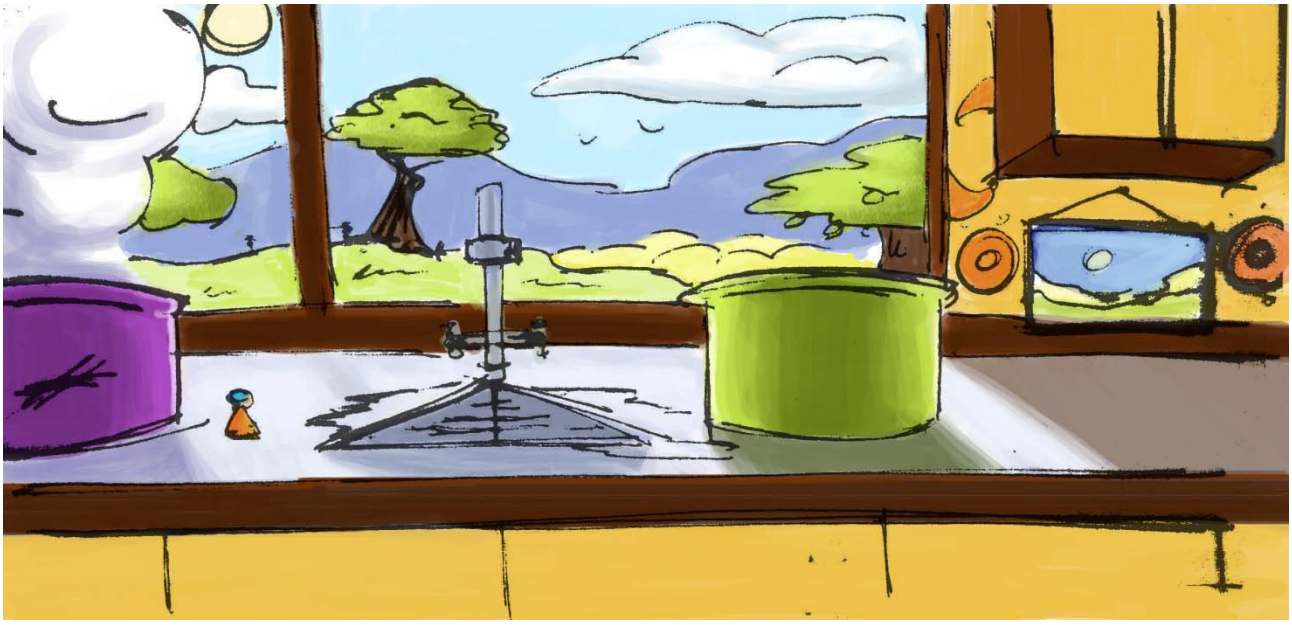
The setting is a kitchen, seen through the eyes of a child, and the main character is the miniature girl Ragga. She is in the middle of constructing a world in flour on the kitchen table, when the radio suddenly sends a storm warning, and the cold wind starts pushing monster big kitchen equipments in her direction. Within minutes, a storm will hit, and Ragga's flour world will be destroyed!

Game Synopsis

The main character is trying to close the window in the kitchen, before the strong wind blowing in destroys the flour world she's building. To do this, she must travel through a windy kitchen environment, where obstacles are blowing towards her constantly. The core play mechanic to do this, is to acquire abilities, switch between them, and use them to jump over obstacles, climb walls and float with the wind.

Setting

The game is set in the house of a chaotic inventor, more specifically in a messy, oversized kitchen. We are seeing the world through the eyes of a child, and everything is huge compared to the main character.



Concept art of the kitchen. Notice the character to the left in the picture.

The chaotic inventor is not aware of anything else than what goes on in his own head. His kitchen is full of stuff that he just needed for something, left on the kitchen table, and forgot about again.

He does not notice the universe that Ragga is building up, which is also the reason why he doesn't realize the dramatic consequences of him opening up the window.

There are dangerous obstacles blowing around with the wind.

Audio-visual Style

Below are some keywords and sentences that describe the audio-visual style of the game:

Steam punk with a touch of Alfons Aaberg and Persepolis

Steam punk: The crazy inventor is one reason for choosing this setting.

A bizarre and abstract style that gives the image of how the main character Ragga looks at things.

Persepolis: Comic inspired universe. Simplicity, flatness.

Alfons Aaberg: poetic fantasy creation.

Game links: little big planet, the scrolling with the character, fantasy universe. Small character, oversized world, abstract meeting with things.

The music style should have some kind of retro, venture, magical and fantasy in it - reflecting the visuals 60 retro style.

Key Features

- Emphasis on wind physics, which functions both as a hindrance and a help to the player.
- Unique ability set, based on household objects: scissor legs, corkscrew bouncy body and brushy afro.
- Over-dimensioned and surreal world, as seen through the eyes of a child.

Target Audience

The target audience for the game are 8-year olds of any gender, but primarily female.

On the story side, we are emphasizing this target group by using a cute main character, and a simple, uncomplicated story. Furthermore, the main character is a girl who the target group can easily sympathize with – a small girl in a big world, but even so, a strongwilled, energetic and optimistic character.

On the gameplay side, we are emphasizing the target group by using a fairly simple gameplay goal – reaching the end of the level. The game is timed, to provide a natural gameplay motivation for completing the level, and the game is quite “forgiving” in its challenge system. Dying will not cause game-over, but simply return the player to the last checkpoint with recharged health. This makes the game less potentially frustrating, and opens up to a more reckless, free mode of play (within the time constraint).

Characters

Ragga

The main character is a miniature girl by the name of Ragga. She is roughly the size of a matchbox. Brave and curious, with a strong will and a funny mind. This is what makes her look at the scissor for instance, and imagine a way of walking differently. Her background for this way of looking at things is of course a consequence of her living with the professor.

During the game, she gets abilities. Her legs can change in to scissor-legs. Her hair can fold out in an amazing afro hair-do, and her torso can screw into the jumping spring.

Ragga in scissor mode

Ragga, floating around in afro mode

Ragga using her spring ability

The Inventor

The chaotic inventor is a secondary character, not really existing in the game. But he is the one that has influenced the girl indirectly through his behavior.

Story

Synopsis

Below is an outline of the story progression in the game.

Intro Cut-scene

The game starts with an introduction cut sequence. Like an animatic, using the comic language. It will be made like a movie clip, not using the in-game graphics.

1. Ragga builds a flour castle on the kitchen table. In the picture is a radio and Ragga in the flour castle. From the radio comes a storm warning.

2. Zoom in on Ragga. Scared, surprised facial expression. Thinking bubble showing her fear of how the storm will destroy the castle.

3. Facial expression changes. She has found her inner fighting power-girl. She looks convinced and ready to "go for it".

4. Thinking bubble with text. " I need to go and close the window before the storm destroys my castle!"

5. Ragga looks around the room for something to help her. She sees a corkscrew, and is inspired by its shape to magically transform her body in a puff of smoke (enters spring mode).

6. She looks determinedly towards the far-away window.

In-game Synopsis

During the game, Ragga progresses further and further towards the right of the level, and finally reaches the window, which she closes by jumping on it (and the window slides down).

Along the way, the radio transmits short messages, stating that the storm is coming closer or similar foreboding messages. These are activated at every major challenge of the game.

The radio. "A storm is coming!"

Gameplay

Overview

The game is a side-scrolling 2½D platformer, where the main character can switch between ability sets which allow her to navigate through the environment.

In this game, the main character must move from the start position, and to the window at the end of the level, in order to close it. To do this, the main character must utilize three abilities – jumping (spring mode), walking on walls (scissor mode) and floating in the air (afro mode). She can also walk firmly attached to the ground, to cope with strong wind resistance (scissor mode). Scissor mode and afro mode must be gathered within the level.

Since the window is open, there is a soft wind blowing against the main character at all times. Because of this, various items in the messy kitchen are blowing towards her, and she must avoid them, for instance by jumping over them. Along the way there are also pitfalls which she must avoid, and other dangers. The game difficulty increases as the character progresses.

At the end of the level, the player must close the window by jumping onto the handle of the window.

Level structure

The ultimate goal of the character is to reach the end of the level and close the window. Along the way, the character must acquire two abilities, which count as subsidiary goals. These will recharge the characters health, and function as checkpoints.

Gradually Increasing Difficulty

As the character progresses through the level, the wind appears to get stronger. This does not affect the character directly, but the objects that are being blown around will become faster and bigger as the game progresses.

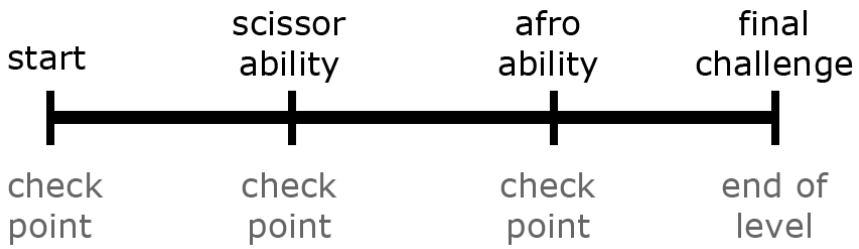
Time Limit

The game has a time limit of 3 minutes. When the time runs out, the game is over. A “game over” screen is displayed, and the player is returned to the menu screen. The game timer is constantly visible on the GUI.

Checkpoint System

The beginning of the level, and each point where the character acquires an ability, serves as a checkpoint. If the character loses all her health, she will respawn at the most recent checkpoint.

Note that obstacles are reset when the character dies and returns to a checkpoint. Obstacles which have already moved are returned to their original positions, so the challenge is the same for each area every time.



Above is a timeline of how the level is built, with regard to checkpoints and the sequence of abilities.

Obstacles

There are a number of different obstacles in the game. Some are relatively harmless, some cause damage, and some are lethal. Their types are detailed below:

Strong Wind

Strong wind prevents the character from moving normally against the wind. It speeds up movement in the direction of the wind. Walking normally (in afro or spring mode) against the wind is not possible – the character will be pushed back.
Once in scissor mode, the character can walk unhindered in the direction against the wind, and will not walk faster in the direction of the wind.

Lose Health

Some obstacles cause the character to lose one point of health:

Object Type 1: Sliding Dangerous Obstacle

This type of obstacle is a 3D object which travels in a single direction along the ground, until it hits a wall, where it will stop. It moves at a constant speed, and has no AI. Touching it will result in the loss of 1 health for the character.

If the character collides with the obstacle, the object will continue along it's path unhindered. The character plays an animation of “taking damage”, and becomes temporarily invulnerable so she can move through the obstacle without losing further health.

Object Type 2: Flying Dangerous Obstacle

This type of obstacle is a 3D object which also travels in a single direction, but at a variable speed (no AI). It flies through the air, and ignores any obstacles in its path. Touching it will result in the loss of 1 health.

As with object type 1, if the character collides with the object, it will continue unhindered and the character will become invulnerable for a short while.

Object Type 3: Stationary Dangerous Obstacle

This type of object is stationary, but causes a loss of 1 health if touched. The collision behavior is the same as the other two obstacle types.

Fall Damage

Falling from a height greater than roughly 4 times the characters height will cause the character to lose a health point. Some floors are soft, and will not cause fall damage.

Instant Death

Touching a kill-zone causes instant death and respawn. These kill-zones can be placed around dangerous things (like fire) or at the bottom of pitfalls.

Hi-score

Upon completing the level, the players score is displayed. This score is equal to the time taken to complete the level. In other words - the shorter the time, the better the score.

Avatar Actions and Movement

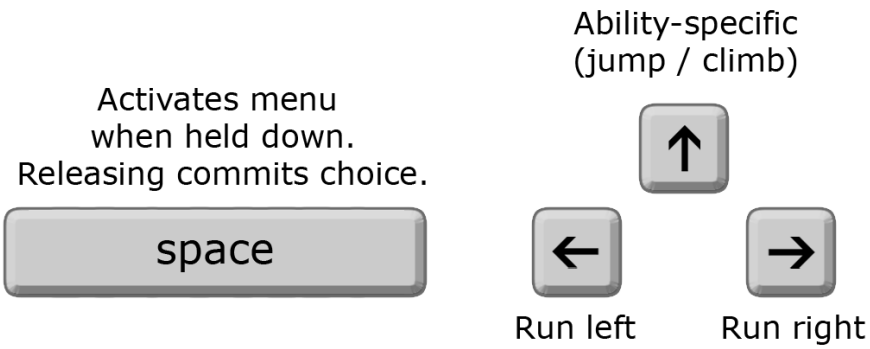
The character moves around in a side-scrolling environment. Along the way she will meet various obstacles, and these obstacles can only be defeated by using her various abilities, for instance by jumping over them, climbing walls or floating with the wind.

Acquiring Abilities

The character starts out with one ability (spring mode), and can acquire a further 2 abilities (scissor mode and afro mode). Once acquired, the player can switch freely between acquired abilities. Acquiring an ability recharges the characters health.

Controls and Control Modes

The basic controls for the game look like this:



As noted, the UP key is ability specific. There are three ability modes in the game. Each ability is tied to a part of the body, and is derived from a kitchen object:

Scissor-legs

Derived from a **pair of scissors**. These legs are pointy and sharp at the end, allowing the character to stand firmly on the ground or even walk vertically on walls. Also protects against strong wind.

Limitations: Only walls (not small objects and obstacles) can be walked. The ceiling cannot be walked on.

Control and motion: Once the character is standing next to a wall, the UP key is pressed. This makes the character stick her leg out quickly, like a stabbing motion, into the wall. She is now stuck to the wall, and can walk vertically instead of horizontally. The same method is used next to the top or bottom of the wall to walk horizontally again. If the UP button is pressed when not next to an edge, the legs stop working, and the character falls. If the player changes abilities, the character is no longer stuck to the wall.

Also note that the character is not affected by wind when using the scissor legs on the ground.

Afro Hair-do

Derived from an **electric shock**. This allows the character to fall more slowly, like a parachute. This is also used to drift in the direction of the wind, while falling slowly (only in windy areas).

Control and motion: In afro mode, the character can jump, but not as high as in spring mode. Roughly the height of the character. The UP key causes the character to jump. The character falls more slowly, at a constant speed (even in non-windy areas). The character can never suffer fall-damage when using afro.

In windy areas, the character will fall at the normal afro-mode pace, but will drift in the direction of the wind at a fast pace (faster than running).

Spring Body

Derived from a **corkscrew**. The torso can be compressed and quickly extended, almost explosively, making the character shoot into the air, like a high jump.

Control and motion: Pressing UP in this form causes the characters to jump. The longer the UP key is held, the farther up the character will jump, to a maximum of roughly 3 times the character's height.

Health

The character has three points of health. A health point is lost when the character comes into contact with a dangerous obstacle. Touching a kill-zone (placed at the bottom of a hole for instance) causes the character to lose all health points.

Once all health is lost, the character respawns at the last checkpoint with full health. Respawning is instantaneous. The game timer is *not* reset.

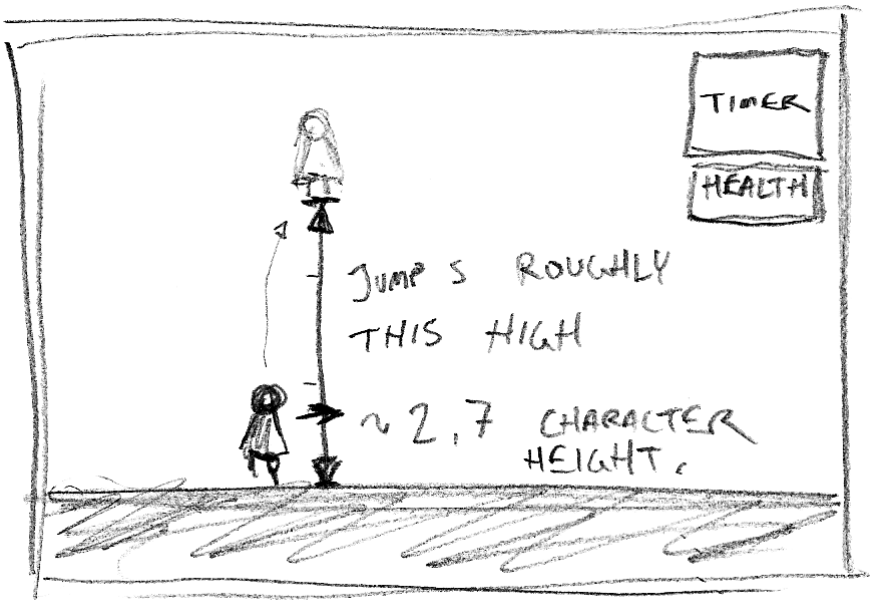
Interface

Menu Screen

The menu screen allows three choices – start the game, credits, and quit.
Starting the game takes the player to the intro cut-scene.

Playscreen Wireframe

Below is a rough wireframe model of the gameplay screen:



The model shows the characters height relative to the screen height, roughly a quarter of the screen height. She can jump slightly less than three times her height in spring mode (roughly half of that in afro mode).

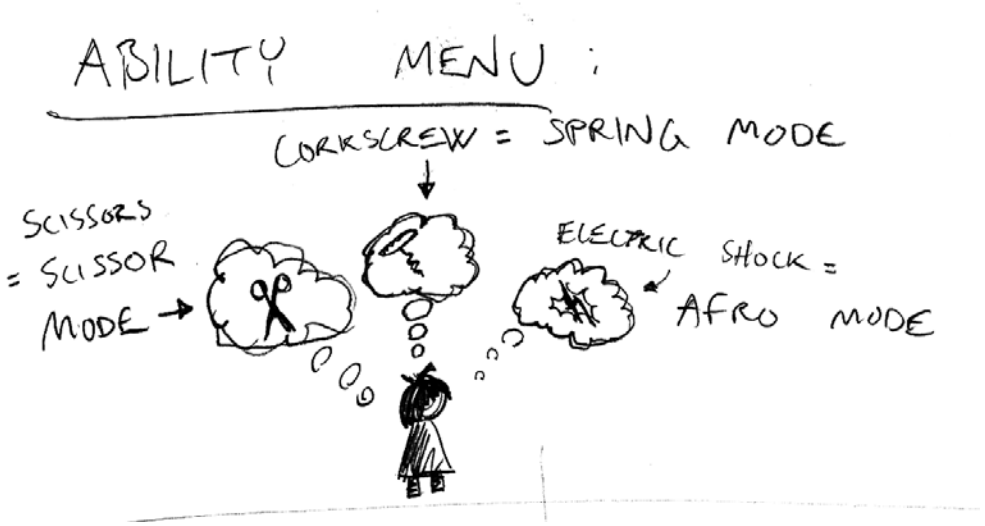
In the corner there is a constant GUI overlay showing the game timer and health.

The camera follows the character, but there is more space in the direction the character is facing. When the character moves, the camera zooms slightly out and shows more of the level in front of the character, relative to the characters speed.

Ability Select Wireframe

Below is a wireframe model illustrating the ability selection GUI. The menu elements are 2D sprites, and are placed in front of all graphics, ie they cannot be blocked by other graphics.

When the SPACE key is held down, a menu appears. Here the player can select between three abilities (UP = spring, LEFT = scissor, RIGHT = afro). See below:

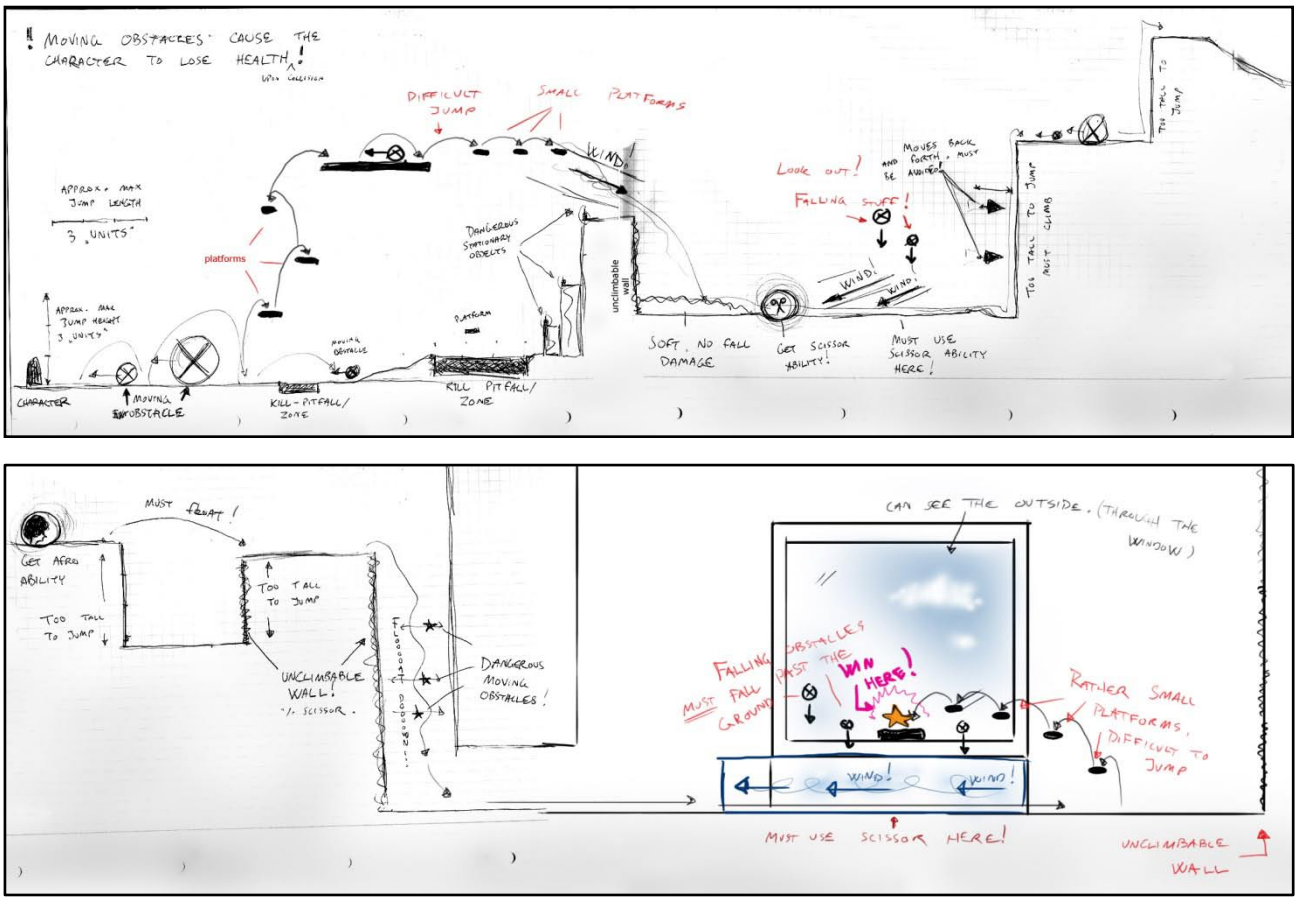


Each think-bubble hovers around a bit, each showing an icon of what it signifies. Scissors for scissor ability, corkscrew for spring ability, an electric shock for afro ability. The selected think-bubble is larger than the others (roughly twice the size). Once SPACE is released, the menu disappears and the choice is committed.

Level Design

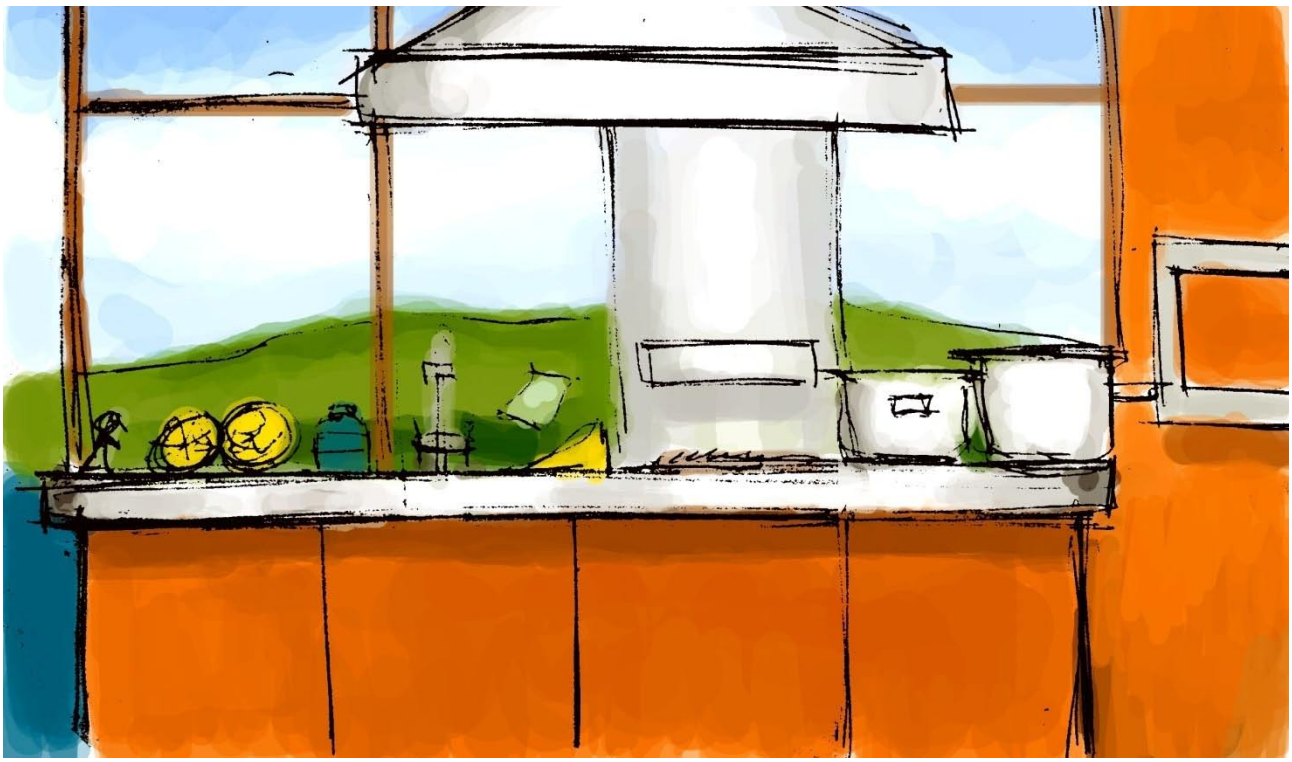
Level 1 – The Dangerous Kitchen

Below is a rough draft of how a level could look, gameplay-wise. The level continues from the right edge of the top picture to the left edge of the bottom picture:



A wireframe model of the gameplay challenges

This was a draft, based on the gameplay challenges. The artists have then put this system of gameplay elements (platforms, obstacles…) in a more graphic representation. Below is a re-rendition of the first 30% of the level:



A more graphic representation of the kitchen level

This is not the final level design, but a draft made to explore what kind of obstacles and challenges the character might face, and how to depict those challenges semi-realistically. Presently, the planned changes to the level design are minimal and corrective (not completely redesigning) in nature.

Challenges in the level include:

- Choosing the most efficient routes through the level
- Precision jumping over moving and stationary obstacles
- Avoiding moving obstacles
- Climbing walls in bursts of movement, to avoid moving obstacles
- Floating across long distances with the afro ability
- Floating down shafts slowly, to avoid enemies suspended in mid-air

At the end of the level the character must pass the open window using her scissor legs to withstand the strong wind. She then jumps up via platforms, to finally jump onto the handle of the sliding window and close it.

Sound Design

This section will explain the role of sound-effects and music in the game, aesthetic as well as technical. More specifically, a documentation of the overall audio design, and how it should be implemented in the game engine Source.

The sound design in the game audio covers the global kitchen universe as well as the gameplay, music and intro sequence.

Kitchen Universe

A soundscape containing typical sounds that normally are associated with a kitchen are to be developed. The sounds don't necessarily need to be in the picture. Sound that is heard outside the picture aims to expand the size of the room. That way, the player will get the feel of the size of the game universe which is important since it shouldn't come as a total surprise to the player when moving objects are coming in from outside the picture.

Gameplay

The game character is moving through a universe which involves danger. This kind of danger is not to be taken too seriously, since this is not a violent action game. The sound effects should reflect that by adding suspense, drama and at the same time there should be some kind of fun to it all.

The main character has her own unique personality (power girl) - the way she moves and transforms etc. The sounds related to her should reflect that.

The sound design also aims to make it clear to the player when bonuses are achieved and the game is won. It should also provide some kind of response/signal to the characters action - informing the player of what is going on and making it understandable what actions are good or bad.

Naturally the intro sequence will use the same sounds as the gameplay.

In general, it is important that all of the sounds are designed to work within a non-linear context - the same sounds may be triggered several times and could easily become annoying in the long run. Therefore the sound should be short, only providing what is needed whenever it is possible. A single sound should work among others.

The Music

We aim to compose music that has themes connected to the obstacles in the gameplay. The idea is some kind of an adaptive soundtrack. The music will consist of a basic loop, that will run through the entire game. There will be four thematic loops that will play when triggered in the gameplay. Once a new loop is triggered, the previously loop should stop.

When the player has won the game there should be another piece of music which reflects that.

Technical Implementation

The technical implementation of the audio in the game is done through a scripting system that communicates with entities in Source. Sound in the game can be implemented in three ways. It could be through the soundscape entity which includes random file selection and pitch, and has variable distribution in the environment – from specific position to everywhere present in the environment. The ambient generic entity has less properties than the soundscape entity, but it is useful when only single sounds should be played once a trigger is hit.

For more advanced sound events (when a character jumps or flies) a new sound entity is aimed to be developed by the teams programmers. The idea is to let the game system sends a call, when triggered, to an entity (simple play specific sound on a call from system) which the team's audio designer can use at script level.

Technical Specifications

The technical specifications are used as a blue-print for the programming team and will be of a technical nature describing how the functional requirements are to be implemented.

Platform and OS

The game is intended to run on a PC computer running Microsoft Windows, any XP or Vista version. The minimum system requirement are: Pentium 4 or better, 2GB ram, 1GB hard drive space, graphics card with Shader Model 3 support.

There is no intention of porting to other platform.

External Code

The game will use the Source SDK game engine used in such titles as Half Life 2, Counter-Strike: Source, etc. The engine was developed for first person shooter (FPS) games but can be modified to other genres. The game will be a modification of the Half Life 2 source code, as this will provide a lot of pre-made systems, such as animation, collision detection, physics, movement etc. The extent of features reusable in Half Life 2 is not completely known at the time of writing.

As the game relies heavily on Half Life 2, the main task of the programming team is to modify Half Life 2 code to fit with the functional requirements. Source and Half Life 2 documentation is scarce, therefore it is difficult to create class and sequence diagrams. Instead a task-list is created which will list the required modification tasks needed to fulfill the functional requirements.

The coordinate system is a right-handed coordinate system with Y pointing into the scene.

Modifications

This is a prioritized list of tasks that need to be completed. Each set of tasks represents a milestone. It is organized based on risk in that core mechanics come before esthetics.

Milestone 1: Playable prototype

- 3rd person camera perspective
  Modify the camera's behavior to view avatar from the side always viewing down Y axis and moving along XZ plane at a configurable distance.
- Graphics pipeline
  Set up the graphics pipeline for visuals and animation group, so they can import new models without programmer assistance. This includes lighting, models, animation, material and relevant properties for these.
- Collision Detection
  Implement collision detection and response and health cost between avatar and flying entities, kill-zones and too high falls (4 times avatar height.)
- Static Objects
  Object does not move, causes 1 hit-point. Avatar can walk through object.
- Jumping Ability (default)
  Implement the movement behavior and characteristics of the avatar as specified in functional requirements in the Corkscrew-ability mode. The longer the UP key is pressed, the higher the

character jumps. This is done by marking time UP is pressed, and adding force in positive Z (up) direction until jump time threshold is reached. Avatar can jump roughly 3 times avatar's height.

- Afro-hair Ability
  Implement the movement behavior and characteristics of avatar in the Afro-hair mode. This mode changes gravity's effect on player and increases speed of avatar inside air-flow entities in direction specified by air-flow entity. Whether avatar can walk against airflow will be controlled by air-flow entity wind speed parameter. Avatar can jump roughly 1.5 times avatar's height. See Jumping Ability for more info on characteristics.
- Air-flow Entity
  Level designer needs an entity (Brush) which represents a strong wind current. Wind current should affect avatar (contrary to wind entity already in Half Life 2). Level designer needs access to wind-direction and wind-speed. Entity is invisible. This task does not actually implement avatar behavior, it merely sends an input to avatar about the zone.
- HUD Elements
  Display 2D graphics on screen on top of play world as an overlay to show things such as time, ability selection menu etc.
- Ability Menu
  Player needs the ability to switch between her avatar's abilities in-game. Each menu-element is a think bubble, which means a HUD element. The menu appears by pressing and holding space, and using arrow keys to select. Game does not pause when in ability menu. Currently selected ability is in double size.
    - Issues: The extent of more work required, should game-play require game paused during ability selection, is unknown.
    - Possible solutions: Hard work and dedication.
- Health Bar
  Display avatar's health-points.
- Kill-zone Entity
  Level designer needs an entity (Brush) which kills the player when touched.

Milestone 2: Beta version

- Scissor-legs Ability
  Implement the movement behavior and characteristics of the avatar in the Scissor-legs mode, walking fixed on scissor-legs-walk-able entities, both horizontally and vertically.
    - Issues: Avatar might need to switch between walking horizontally and vertically if level so requires. The player class needs to be able to switch from being "connected" to one walk-able entity to another. Avatar will also need to connect to vertically walk-able entity before actually standing on entity.
    - Possible solutions: Ray casting to find closest walk-able entity. If in range, move avatar there.
- Scissor-legs Walk-able Entity
  Level designer needs an entity (Brush) which affects the player when scissor-legs are enabled. The entity makes the player progress at a fixed pace unaffected by air-flow entity. If entity placed vertically, avatar will walk vertically. Avatar behavior not included in this task.
    - Alternatives: Just make a trigger area and connect to correct input on avatar to signal when in this area type.
- Title Menu
- Intro sequence

- Game-over sequence
- Level Checkpoints
  Implement level check-point which will restore avatar health, add specific ability and store location.
- Reset
  Level reset when avatar looses all health (hit-points.) Re-spawn at last recorded checkpoint. All level objects need to be reset, re-position avatar at last reached check-point, and restore avatar's health. Timer is not reset.
    - Issues: At time of writing it is not known how this could be done.
    - Possible solutions: Save game (excl. time elapsed) when check-point reached, and re-load when avatar dies.
    - Alternatives: Similar functionality is already in Half Life 2, so finding how these check-points are implemented and adapting to our needs is a possible solution.
- Soft Ground Entity
  Implement a soft-ground entity so level designer can place an invisible brush just above level ground which will slow down the avatar before it hits the ground. Must be slow enough to not give visual clues to this entity being present.
    - Alternative: Use ray-casting to determine when to slow avatar down just before hitting ground. This will allow entity to be within the ground entity. Another alternative is to trigger avatar's invulnerability for a few milliseconds to allow him to land without damage.
- Flying Objects
  Objects that should damage avatar if colliding by 1 hit-point. Objects fly from right to left, but should increase in speed as player progresses through level. It therefore needs an adjustable speed parameter. Cannot be affected by avatar, i.e. it will continue unhindered if colliding with avatar.
- Sliding Objects
  Objects that move along the surface of the level which damage the avatar with 1 hit-point if colliding. Should increase in speed as the player progresses through level. It therefore needs an adjustable speed parameter. Object cannot be affected by avatar.
- Invulnerable State
  Implement a invulnerable state on avatar, set after a collision which will switch back to normal state after specific amount of time (adjustable by game designer.)

Milestone 3: Play-test version

- Avatar Transformations
  Implement model changing according to change in ability. Add needed animations not already available in Half Life 2, I.e. switch from horizontal walk to vertical walk and back in scissor-leg mode, hovering animation in afro-hair mode, harmonica animation, left-to-right turn and right-to-left turn.
    - Issues: The spaghetti-dish that is the Source source-code.
    - Possible solutions: Liaise with other groups needing more animation functionality.
- Timer
  A timer needs to count down from 3 minutes and end game when time runs out. This is a visual animation displayed in upper right corner.

Milestone 4: Final Release

- Transition Cloud
  Implement a particle system which can be triggered at character transitions above character.

- Predictable Camera
  The camera must zoom out a little when avatar moves camera should show more of level in direction of movement, possibly by using motion vector as target view point instead of avatar. Distance camera so avatar is ¼ screen height.
    - Issues: The camera is a client-side object, while the player (avatar) is server-side. The player's speed value is sent to the client at regular intervals, and therefore the camera movement will be jerky.
    - Possible solutions: An exponential filter on the position and speed variable on client-side.
- Afro-hairdo Shader
  Avatar's hair should be affected by wind and movement to give a better physical feel.
    - Issues: Avatar movement and wind information needs to be sent to shader program. Team's shader knowledge in Source is limited to none.
    - Possible solutions: Liaise with other teams who might have shader experience in Source.
    - Alternatives: Rig hair to jiggle-bones (believed to be for a similar purpose) in Source or create it as animation.
- Credit sequence
- Win sequence
- Animated Ability Menu
  Think bubbles need to move around, either with some randomness or mixture of sinuses on each axis.
- Sound Entity
  Implement a sound entity with signals from game, such as character jumping, flying etc. which can be used by audio at scripting level.
- Hi-score
  Show time elapsed at end of level.

Other Elements

Other information, such as flow of data and main control loop are not considered relevant to describe as these are heavily re-used from the Half Life 2 build, where most of the needed features, e.g. health, already are implemented.

Artificial Intelligence

There are no NPCs capable of any intelligent action. Still, the flying obstacles are considered and treated as NPCs and require coding. There are three types of objects, flying, moving on ground and static.

All objects (except static) need to be triggered when avatar is close to start them moving leftward direction.

Additional Features

This is a list of features in the game, aside from the ones detailed in the technical specifications. It is not exhaustively specific, but is used to overview and prioritize the various features of the game. The first points in each category are the highest priorities.

Audio-visual Features

- Call / trigger sound system (scripting tool)
- Dynamic camera (zooms out and moves relative to characters speed)
- Adaptive soundtrack

Non-interactive sequences

- Intro sequence
- Credits

Character models

- Ragga
    - Spring mode
    - Scissor mode
    - Afro mode

Cut-scenes

- Intro sequence
- Time-out sequence

Project Estimate (first draft)

Here is the first draft of the estimates and risk management for the project. Unfortunately the animation estimates haven't been done soon enough to be included.

The Brushy WindBlown Adventure

Platform PC
Ship Date: 27.May 2008
Programming:

Game System	Task Description	Best	Worst	Most Likely	Result
Camera	3rd person camera perspective	1	7	2	4
Camera	Predictable Camera	0,5	2	1	1
Pipeline	Graphics pipeline				0
Sequences	Win sequence	0,5	2	1	1
Sequences	Intro sequence	0,5	2	1	1
Sequences	Game-over sequence	0,5	2	0,5	1
Sequences	Credit sequence	0,5	2	0,5	1
Menu	Title Menu	0,5	2	0,5	1
Grafical Shader	Afro-hairdo Shader	2	7	5	6
GUI element	HUD Elements	1	3	1	2
GUI element	Health Bar	0,5	2	0,5	1
GUI element	Ability Menu	0,5	2	0,5	1
Animation	Animated Ability Menu	0,5	1	0,5	1
Entity	Air-flow Entity	0,5	3	0,5	1
Entity	Kill-zone Entity	0,5	3	0,5	1
Entity	Soft Ground Entity	0,5	3	0,5	1
Entity	Scissor-legs Walk-able Entity	0,5	3	0,5	1
Engine element	Collision Detection	1	5	2	3
Engine element	Invulnerable state	0,5	1	0,5	1
Engine element	Flying Objects	1	4	2	3
Engine element	Sliding Objects	1	7	4	5
Engine element	Static Objects	0,5	2	1	1
Animation	Avatar Transformations	2	7	4	5
Charcter design + engine	Afro-hair Ability	0,5	4	1	2
Charcter design + engine	Jumping Ability (default)	0,5	2	1	1
Charcter design + engine	Scissor-legs Ability	2	7	4	5
Animation	Transition Cloud	0,5	3	1	2
Level element	Level Checkpoints	1	4	2	3
Engine element	Reset	1	7	3	5
GUI element	Timer	0,5	2	0,5	1
Sound	Sound Entity	1	5	2	3
	High Score	0,5	1	0,5	1
Audiodesign					67

Game System	Task Description	Best	Worst	Most Likely	Result
	Soundeffect	5	10	7	8
	Musik	2	4	3	3
	Speak	0,5	1	1	1